CSC34300 (Spring 2018)
Lab 04: MIPS Assembly Language
(8 points)

IMPORTANT!

Please follow the **submission guidelines** below or your submission will be rejected.

1. You are expected to submit both the lab report and the MIPS assembly source files to Blackboard in a single submission attempt.
2. The MIPS assembly source files should be put into a single ZIP file.
3. Your report and the ZIP file need to be named as follows
    1) The name of your project report should be "FirstName_LastName_Lab_**XX**_Report.pdf". Replace "XX" with the actual project number (2 digits). Note that your project can be a WORD file or a PDF file.
    2) The name of your project file should be "FirstName_LastName_Lab_**XX**_Source.zip". Replace "XX" with the actual project number (2 digits).

In this lab, the students are expected to develop MIPS assembly programs in MARS, an interactive development environment (IDE) for programming in MIPS assembly language.

MARS has been installed on the class virtual machine. You can also download a copy of MARS from the Internet (http://courses.missouristate.edu/kenvollmar/mars) and install it on your own computer.

Read the assignment carefully, as you should do for every assignment.

Task 3: Fibonacci Number and Recursive Function (4 points)

Objectives:

You will practice recursive function calls in MIPS assembly code. Your code will repeatedly read an integer *n* from the console, compute the Fibonacci number F(*n*) and print F(*n*) to the console (followed by a new line). If *n* is too large (i.e. causes overflow) which is detected in the function that computes the Fibonacci number, your code prints "The number is too large." The program terminates if $n = 0$.

Expected outputs:

```
CSC34300: Your Name and ID
Please enter a Fibonacci index number (0 will stop the program):10
F(10)=55
Please enter a Fibonacci index number (0 will stop the program):46
F(46)=1836311903
Please enter a Fibonacci index number (0 will stop the program):47
F(47)=The number is too large
Please enter a Fibonacci index number (0 will stop the program):0
bye.
```

Description:

Use system call 5 to read an integer from the console. The returned value is in $v0.

You code will compute Fibonacci number recursively. The function interface looks like

```
int64  Fibonacci2 (int n)
```

Different from previous labs, `Fibonacci2` calculates the Fibonacci sequence in a recursive way. The function takes one integer as input and returns F(*n*) and F(n – 1), two 32-bit integer numbers (thus int64). The argument is in $a0. F(*n*) is placed in $v0, and F($n$ – 1) is in $v1. Since you have both F(*n*) and F(n – 1) with a function call, you can easily compute F(n + 1). The pseudo code looks like

```
// $a0 is n
Allocate space on stack and save registers
If ($a0 == 0)
      $v0 = 0 and $v1 = 0
Else if ($a0 == 1)
      $v0 = 1 and $v1 = 0
Else if ($a0 > 100)
```

```
            $v0 = 0xFFFFFFFF
      Else
            Decrement $a0
            Call Fibonacci2 to get F(n-1) and F(n-2)
            If ($v0 is not 0xFFFFFFFF)
                  Compute F(n)
                  If (no overflow)
                        Set correct values in $v0 and $v1
                  Else
                        $v0 = 0xFFFFFFFF
      Restore saved registers and stack
      Return
```

Please pay attention that in the pseudo code only one recursive call is made in `Fibonacci2`, and implement it in MIPS assembly accordingly, otherwise your program will hang when *n* is big.

The main function of your code will call `Fibonacci2` with one argument. Your code needs to preserve all the registers changed, except for $v0 and $v1.

Requirements:

1. Your main program should
   a. Let user input the index number *n*.
   b. Calculate the Fibonacci number corresponding to the index number *n*.
   c. Repeatedly take user inputs. When the input *n*=0, the program stops.
   d. The calculation should be completed in reasonable amount of time for all index numbers, i.e. you program should not hang.
   e. Must be a recursive function.
   f. If overflow is detected during the calculation, prompt "The number is too large."
2. You need to comment **EACH** line of your MIPS assembly codes.
3. List the registers usage of your function(s) in the source code as comments, before the function implementation starts.

Note:

1. Your program should follow proper function calling steps.
2. Assume all numbers are signed.
3. Do not use instruction `add` or `addi`. Use `addu` and `addiu` instead.

Deliverables

1. Your MIPS program, named as `FirstName_LastName_Lab_4.asm`, should
   1) Meet all requirements.
   2) No compilation errors from the assembler.
   3) Run correctly in MARS.
   4) Show the expected results.
   5) Have proper amount of comments (see requirements).
   6) Be in a ZIP file (see submission guidelines)
2. In your report
   1) Include your entire source codes.
   2) Include a screenshot of your program outputs.